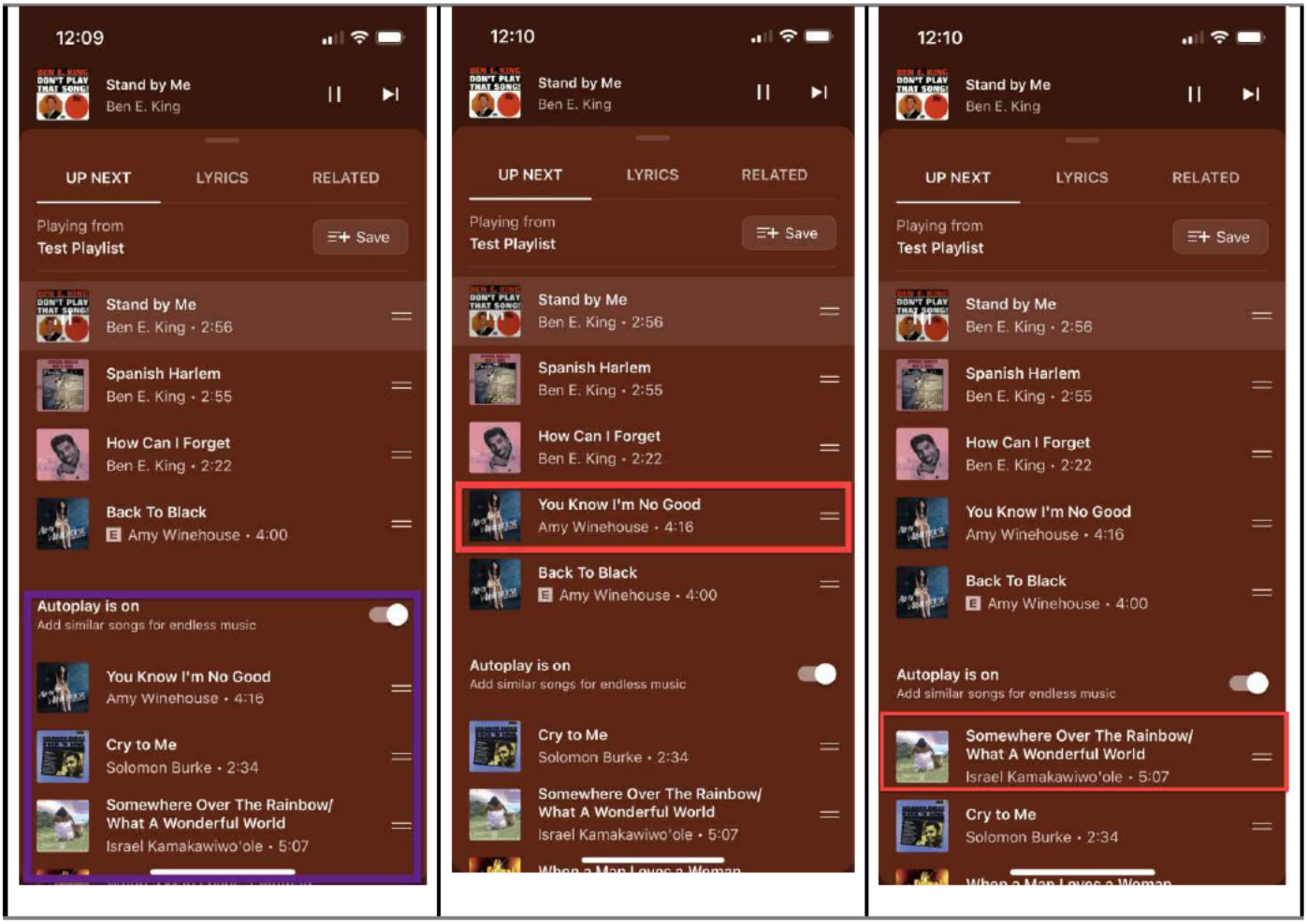
EXHIBIT 21
Filed Under Seal

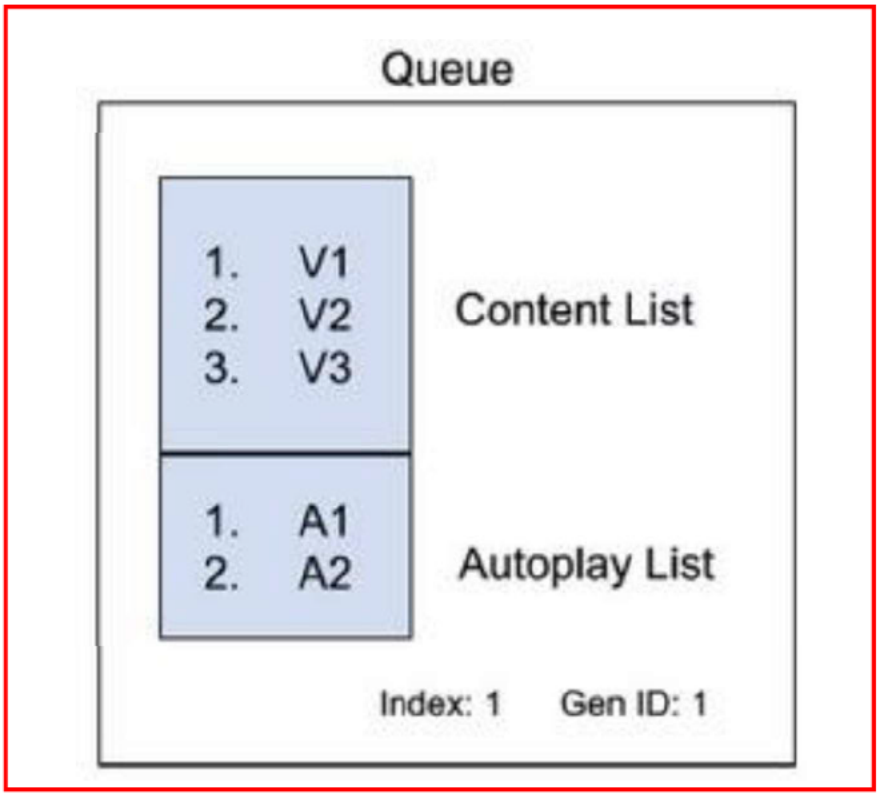


65. Continuing with the above example and with reference to source code, the file PlaybackQueue.java specifies the queue interface and the file DefaultPlaybackQueue.java[3] implements the queue. They maintain locally, on the mobile device, a user selected list ("QUEUE_LIST") that is the section appearing above the "Autoplay" toggle button. If Autoplay is enabled, they also maintain an Autoplay List with recommended media items ("AUTONAV_LIST") that is the section that appears below the Autoplay toggle button. GOOG-SONOSNDCA-00073395 ("The Queue has two lists, one is user editable and the other is server driven."). In the example above the user selected list is initially populated with a User Playlist,

[3] *See, e.g.*, 2020-10-08-youtube_music_3.87.53/java/com/google/android/libraries/youtube/player/features/queue/

but it may also be populated with other media items and playlists (e.g., an album playlist). *Id.*; *see also* GOOG-SONOSWDTX-00041618 ("A music container entity (i.e., album, radio) is added to a local queue when users play it locally… entity information is kept in the local queue…").

66. When a user begins playback of the User Playlist, the mobile device populates its local playback queue with the media item in the User Playlist. It also sends a WatchNext request and receives a WatchNext response containing Autoplay media items that it stores in its local playback queue. The user selected and Autoplay lists are stored locally on the sender device as Android SparseArrays. A high-level illustration of the Playback Queue on the sender device is shown below:



GOOG-SONOSNDCA-00073395 (The Language of Queues).

67. The mobile device plays back the local playback queue. For instance, in the above example the user selected portion of the queue is initially populated with three songs ("Stand by Me", "Spanish Harlem," and "How Can I Forget") from the "Test Playlist," and then Autoplay

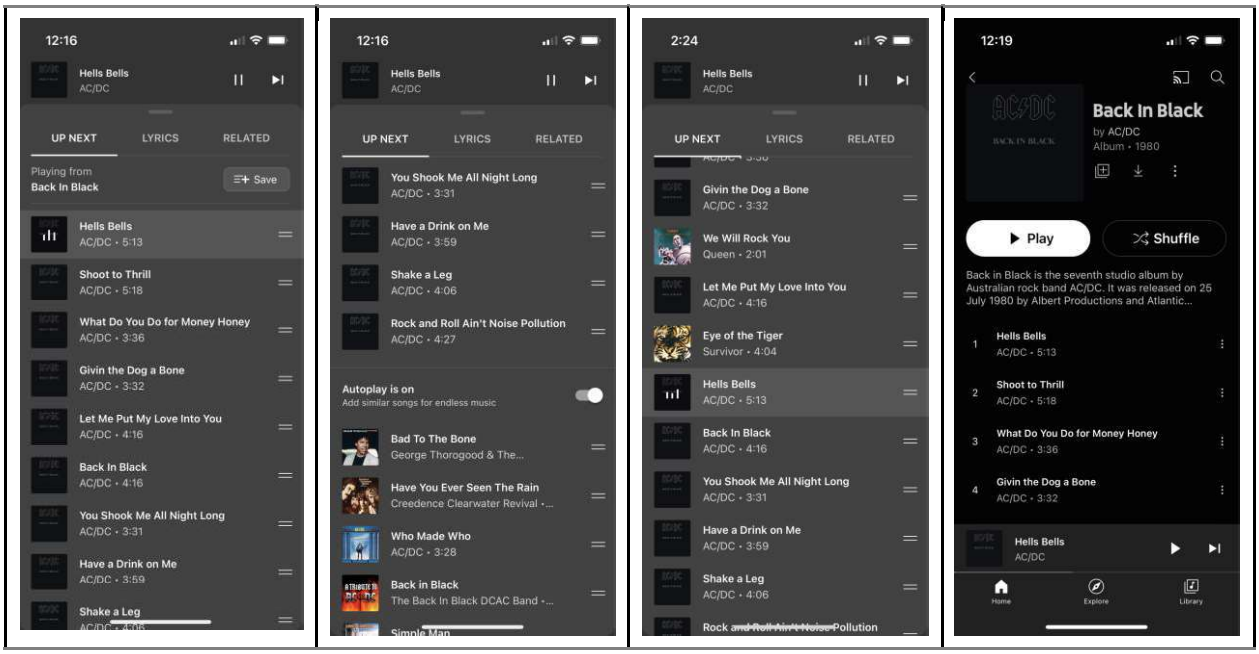
List is populated with the additional service-recommended songs (e.g., e.g., "Put Your Head on My Shoulder," "Johnny B. Goode," etc.). After the mobile device exhausts playback of the user selected list, the first media item in the Autoplay list is added to the user selected list and selected for playback. A user may also select an Autoplay List item at any time, in which case the selected Autoplay List item (and all preceding items in the Autoplay List) are moved to the QUEUE_LIST and the selected item will be played back. For instance, if the user selects "Johnny B. Goode," then that item and the previous item in the Autoplay List (i.e., "Put Your Head on My Shoulder") are added to the "QUEUE_LIST" on the mobile device and "Johnny B. Goode" will be played back.

68. Continuing with the above example, the YouTube Music application allows media items in the user selected list (QUEUE_LIST) and Autoplay List (AUTONAV_LIST) to be edited by a user (e.g., reordered, removed, or moved from the Autoplay List to the user selected list) on the mobile device. DefaultPlaybackQueue.java, lines 296-323. For instance, I showed above examples in which media items were added to the queue and media items in the queue were reordered. *See supra* ¶¶62-64. These manipulations are entirely local and are not based on inputs by a cloud server, and, indeed, are not reflected back to any so-called queue on a YouTube server. *See* DefaultPlaybackQueue.java, lines 296-323. I note that the YouTube Main application allows a user to edit a user playlist, but does not allow a user to edit service-recommended media items.

69. If a user selects the album playlist AC/DC Back in Black using the YouTube Music application on their phone, and then subsequently edits the playlist by, for instance, reordering the songs in the album, the user's phone will play back the songs in the new order. Notably, the version of the AC/DC Back in Black album playlist that is stored in the servers does not change to reflect the reordering of the songs (it will remain in the same order). The fact that the user's phone plays back the songs in the new order despite the order on the servers remaining the same

demonstrates that the user's phone is playing back a local queue that is merely populated with songs from the AC/DC Back in Black album.

70. In the image on the left the contents of the AC/DC Back in Black album have been added to the queue and the first song in the playlist "Hells Bells" is playing back. In the second image from the left Autoplay has been enabled and the queue has been populated with additional media items. In the second image from the right the queue has been reordered, including by moving songs from the AC/DC Back in Black album above "Hells Bells" and moving Autoplay media items from the Autoplay list to the user selected list. Because the local playback queue and the playlist from which it was initially populated are two distinct items, changes to the local playback queue are not reflected in any playlist. Indeed, the AC/DC Back in Black album playlist continues to include the original set of songs in the original order, as shown in the image on the far right.



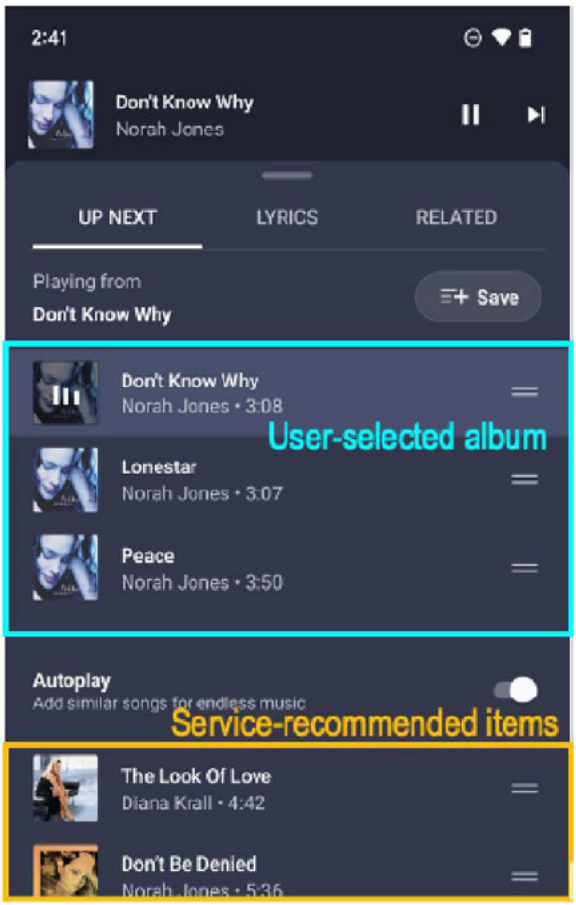
"computing device") uses a "*local* queue" when not Casting. For instance, Dr. Schmidt states that "the Sender's local queue is often loaded with (i) one or more 'videoIds' for the initial user-selected media item or collection of media items and (ii) one or more videoIds for service-recommended media item(s) seeded by the initial user selection." Schmidt Rpt., ¶129. He further states that "[a] representation of the Sender's local queue loaded with data identifying one or more media items from the Watch Next queue is typically displayed by the Sender." *Id.*, ¶130; *see also id.*, ¶132 ("selecting the skip forward transport control causes the Sender to begin rendering the next media item of its local queue.).

166. The documents Dr. Schmidt cites in his report also state that a User Device plays back a "local" queue when not Casting. As one example, Dr. Schmidt cites to a document called "The Queue." GOOG-SONOSWDTX-00039798 ("The Queue"). This document explains that when not Casting "[t]he YouTube Music queue is a primarily a client-side construct." The document further explains that "YouTube Music 1st party clients persist a list of tracks represented by PlaylistPanelVideoRenderer message," and that "queue actions, such as removing items, re-ordering, or shuffle/repeat are performed as manipulation on the client-persisted list of renderers." *Id.* Notably, the document repeatedly distinguishes the non-Casting use case from the Casting use case, explaining that the queue is local in the former case and remote in the latter case. GOOG-SONOSWDTX-00039798 ("The Queue") at -799 ("YouTube Music clients can Cast to Living Room devices. In comparison to the 1st party case where the queue is a client-side construct, the Casting case stores the queue in YouTube servers as a 'Remote Queue' playlist."), -798 ("When Casting, the queue is persisted as a server-side 'remote queue.'"). In other words, this document discloses that when not Casting the queue is a construct that is local to the YouTube Sender, whereas when Casting the queue is a "server side" construct. As another example, Dr. Schmidt

cites to a document called "YouTube Music Playback History in MDx Proposal." Schmidt Rpt., ¶129 (citing GOOG-SONOSWDTX-00041617). This document explains that album playlists and radio playlists are stored in a "local queue" of the sender device: "[a] music container entity (i.e. album, radio) is added to a local queue when users play it locally on the YTM mobile apps." GOOG-SONOSWDTX-00041617 at -618.

167. Other documents I have reviewed also refer to the queue being local prior to Casting. *See, e.g.,* GOOG-SONOSWDTX-00051490 ("YM Cast: Loop, aka Repeat") at -490 ("YTM iOS, Android, and web clients determine the next track to play by persisting and managing the queue state locally."); GOOG-SONOSWDTX-00042754 ("Orbit Queue") at -755 ("MdxLocalPlaybackQueueDataExtractor is the class in charge of providing the current contents of the local queue as well as the playback position so that we have it when we launch the YouTube application on the TV to have the first sync up.").

168. Similarly, in connection with the Patent Showdown on the '615 patent Dr. Schmidt repeatedly stated that a User Device plays back a "local" queue in the non-Casting state. For instance, the quote on the left is from Dr. Schmidt's declaration, and the image on the right is one he provided to show an "example" of the "Sender's local queue":

A YouTube Sender can operate in a non-Casting state in which the Sender itself plays back media… While in the non-Casting state, each YouTube app running on a Sender enables a user to select a single media item (e.g., a song or video) for playback at the Sender, and each YouTube app except for the YouTube TV app allows a user to select a collection of media items (e.g., an album or playlist) for playback at the Sender. Based on the initial user-selected media item or collection of media items, a Google cloud service will automatically recommend one or more additional media items that should be played by the Sender after the initial media item or collection (for some of the YouTube apps, an "Autoplay" setting must be enabled for this to occur).… **As a result, the Sender's local queue is loaded with one or more "videoIds" for the initial user-selected media item or collection of media items and, at least in some cases, one or more videoIds for the service-recommended media item(s) seeded by the initial user selection**. *See, e.g.*, GOOGSONOSWDTX-00039798, 99 ("The GetMusicWatchNext endpoint provides queue renderers when initiating playback on a new container, on queue continuation loads, and for the autoplay section of the queue."); Nicholson Dep. Tr., 101:3-9 ("[T]here's a video ID in those renderers."). **A representation of the Sender's local queue is typically displayed by the Sender, with one example shown here for YouTube Music.** *See also, e.g.*, GOOG-SONOSWDTX-00039798, 98.	

5-5-2022 Declaration of Douglas Schmidt in Support of Sonos's Motion for Summary Judgment ¶57 ("the Sender has a local queue that syncs with the remote queue when in a Cast session with a Receiver"); 5-5-2022 Sonos Opposition to Google's Motion for Summary Judgment at 3:15-19 ("the Sender has a local queue that syncs with the remote queue when in a Cast session with a Receiver.").

does not expressly mention a "remote playback queue," it discloses that its control device can be used to control a wide variety of "appliances," including "a sound system" and a "stereo" just to name a few. *Id.* at 3:15-26. A person of skill in the art would thus understand that the technologies in this patent could be applied to devices that use "playback queues." Thus, the patent contemplates its invention being applied to the field of media playback and home audio systems.

330. U.S. Patent No. 7,734,286 is also entitled "Remote Control System" and claims priority to the '213 patent. For at least the same reasons that the '213 patent is comparable to the asserted '033 patent, the '286 patent is comparable to the asserted '033 patent.

331. U.S. Patent No. 8,254,901 is also entitled "Remote Control System" and claims priority to the '213 patent. For at least the same reasons that the '213 patent is comparable to the asserted '033 patent, the '901 patent is comparable to the asserted '033 patent.

332. U.S. Patent No. 8,831,584 is entitled "Remote Control System" and claims priority to the '213 patent. For at least the same reasons that the '213 patent is comparable to the asserted '033 patent, the '584 patent is comparable to the asserted '033 patent.

B. The [redacted] Google License Agreement

333. The [redacted] Google License agreement is a license agreement between [redacted] the licensors, and Google, Inc. the licensee, related to routing of data between electronic devices. I understand that on January 4, 2023, Google licensed some of [redacted] patents.

334. I have reviewed the [redacted] patents subject to this agreement and have determined that they are technologically comparable to those asserted by Sonos here. Similarly to the Sonos patents at issue here, the [redacted] patents are directed toward solving similar problems associated with the ability to access data from different devices.